

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-282695

(43)Date of publication of application : 15.10.1999

(51)Int.Cl.

G06F 9/46

G06F 9/46

(21)Application number : 11-035205

(71)Applicant : INTERNATL BUSINESS MACH
CORP <IBM>

(22)Date of filing : 15.02.1999

(72)Inventor : YOCUM PETER B
EILERT CATHERINE K
ARWE JOHN E

(30)Priority

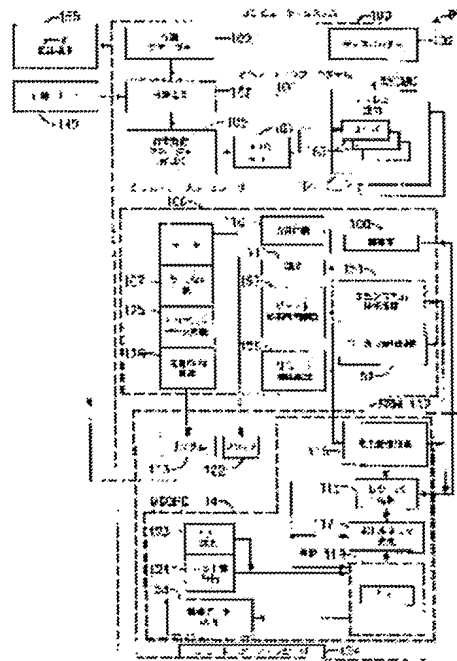
Priority number : 98 38573 Priority date : 18.03.1998 Priority country : US

(54) METHOD AND DEVICE FOR CONTROLLING NUMBER OF SERVERS IN MULTISYSTEM CLUSTER

(57)Abstract:

PROBLEM TO BE SOLVED: To control the number of servers in a multisystem cluster.

SOLUTION: A computer system 100 has a queue 161 for organizing an inputted work request as a service class, transferring respective service classes to a cluster and serving them by servers 163. Prescribed performance indexes are previously allocated to respective service classes. Each system 100 selects a certain service class as a donor class for providing a system resource based on a degree of coincidence of each service class with its own target and selects another service class as a receiver class for receiving the system resource. When a resource bottleneck causing the receiver class to lose its target is the number of servers, each system 100 determines whether several servers are to be added to the receiver class or not based on judgement whether a plus effect of server addition to the performance index of the receiver class exceeds the minus effect applied to the performance index of a donor class or not.



LEGAL STATUS

[Date of request for examination] 09.08.1999

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]	3121584
[Date of registration]	20.10.2000
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平11-282695

(43)公開日 平成11年(1999)10月15日

(51) Int.Cl.⁶

G O 6 F 9/46

識別記号

3 4 0
3 6 0

FI

G O 6 F 9/46

3 4 0 D

360C

審査請求 未請求 請求項の数17 O L (全 20 頁)

(21)出願番号 特願平11-35205

(22)出願日 平成11年(1999)2月15日

(31)優先權主張番号 09/038573

(32)優先日 1998年3月18日

(33)優先權主張国 米国 (US)

(71)出願人 390009531

インターナショナル・ビジネス・マシーンズ・コーポレーション

INTERNATIONAL BUSIN
ESS MASCHINES CORPO
RATION

アメリカ合衆国10504、ニューヨーク州
アーモンク (番地なし)

(72)発明者 ピーター・ビィ・ヨウム

アメリカ合衆国12590、ニューヨーク州ワ
ッピンガーズ・フォールズ、ワイルドウッ
ド・マナー 17ビイ

(74)代理人 弁理士 坂口 博 (外1名)

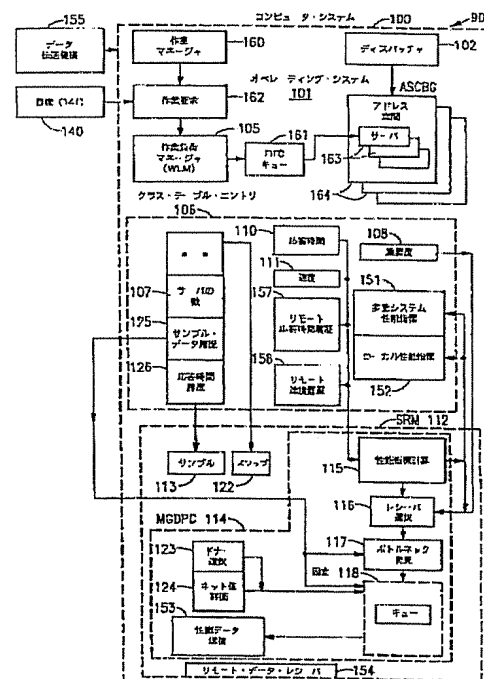
[最終頁に続く](#)

(54) 【発明の名称】 多重システム・クラスタ内のサーバの数を制御する方法及び装置

(57) 【要約】

【課題】 多重システム・クラスタ内のサーバの数を制御する方法及び装置の提供。

【解決手段】 入来作業要求がサービス・クラスに編成され、それらの各々がクラスタに渡って、サーバによりサービスされるキューを有する。各サービス・クラスは予め、所定の性能指標を割当てられる。各システムはサービス・クラスがそれらの目標にどれ程良く合致するかにもとづき、あるサービス・クラスを、システム資源を提供するドナー・クラスとして選択し、別のサービス・クラスを、システム資源を受け取るレシーバ・クラスとして選択する。各システムは次に、レシーバ・クラスがその目標を逸する原因となる資源ボトルネックがサーバの数の場合、各システムは、サーバの追加がレシーバ・クラスの性能指標に及ぼすプラスの効果が、ドナー・クラスの性能指標に及ぼすマイナスの効果を上回るか否かにもとづき、幾つのサーバがレシーバ・クラスに追加されるべきかを決定する。



【特許請求の範囲】

【請求項 1】情報処理システムのクラスタ内においてサーバの数を制御する方法であって、あるサービス・クラスに属する入来作業要求が、前記クラスタ全体に渡るキューに配置され、前記クラスタの前記システム上の 1 つ以上のサーバにより処理されるものにおいて、

1 つ以上のサーバが前記サービス・クラスに追加されるべきか否かを決定するステップと、

1 つ以上のサーバが前記サービス・クラスに追加されるべきと決定される場合、前記サーバが追加されるべき、前記クラスタ内のターゲット・システムを決定するステップと、

前記サーバを前記ターゲット・システム上に追加するステップとを含む、方法。

【請求項 2】前記決定するステップが前記システムの各々により実行される、請求項 1 記載の方法。

【請求項 3】前記システムの各々が、該システムがターゲット・システムか否かを決定し、前記ターゲット・システムであると決定するとき、1 つ以上のサーバを局所的に追加する、請求項 2 記載の方法。

【請求項 4】前記サービス・クラスが第 1 のサービス・クラスであり、前記クラスタが少なくとも 1 つの他のサービス・クラスを有し、前記サービス・クラスの各々が、それぞれに対して定義された性能測定を有する、請求項 1 記載の方法。

【請求項 5】サーバが前記第 1 のサービス・クラスに追加されるべきか否かを決定するステップが、所定数のサーバを前記第 1 のサービス・クラスに追加することが、前記第 1 のサービス・クラスの前記性能測定に及ぼすプラスの効果を決定するステップと、前記所定数のサーバを前記第 1 のサービス・クラスに追加することが、1 つ以上の他のサービス・クラスの前記性能測定に及ぼすマイナスの効果を決定するステップと、

前記第 1 のサービス・クラスの前記性能測定に及ぼすプラスの効果が、1 つ以上の他のサービス・クラスの前記性能測定に及ぼすマイナスの効果を上回るか否かを決定するステップとを含む、請求項 4 記載の方法。

【請求項 6】サーバが追加されるべき、前記クラスタ内のターゲット・システムを決定するステップが、前記クラスタ内の任意のシステムが、1 つ以上の追加のサーバを追加するための十分な遊休容量を有するか否かを決定するステップと、

前記クラスタ内の任意のシステムが、1 つ以上の追加のサーバを追加するための十分な遊休容量を有する場合、1 つのこうしたシステムを前記ターゲット・システムとして選択するステップとを含む、請求項 1 記載の方法。

【請求項 7】最大の未使用容量を有するシステムが前記ターゲット・システムとして選択される、請求項 6 記載の方法。

【請求項 8】サーバが追加されるべき、前記クラスタ内のターゲット・システムを決定するステップが、前記クラスタ内のいずれのシステムも、1 つ以上の追加のサーバを追加するための十分な遊休容量を有さない場合、1 つ以上のサーバの追加が他の作業に最小の影響を及ぼす前記クラスタ内のシステムを決定するステップと、

そのシステムを前記ターゲット・システムとして選択するステップとを含む、請求項 6 記載の方法。

【請求項 9】マシンにより読出され、前記マシンにより実行されて、請求項 1 記載の方法を実行する命令のプログラムを実現する、プログラム記憶装置。

【請求項 10】情報処理システムのクラスタ内においてサーバの数を制御する装置であって、あるサービス・クラスに属する入来作業要求が、前記クラスタ全体に渡るキューに配置され、前記クラスタの前記システム上の 1 つ以上のサーバにより処理されるものにおいて、

1 つ以上のサーバが前記サービス・クラスに追加されるべきか否かを決定する手段と、

1 つ以上のサーバが前記サービス・クラスに追加されるべきと決定される場合、前記サーバが追加されるべき、前記クラスタ内のターゲット・システムを決定する手段と、

前記サーバを前記ターゲット・システム上に追加する手段とを含む、装置。

【請求項 11】情報処理システムのクラスタ内において、キュー内の各作業要求を処理可能なサーバの可用性を保証する方法であって、入来作業要求が前記キューに配置され、前記システム上の 1 つ以上のサーバにより処理されるものにおいて、

前記キューに対してサーバを有さない前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在するか否かを決定するステップと、

前記キューに対してサーバを有さない前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在すると決定された場合、前記作業要求が親和性を有する前記サブセット内のシステム上で、前記キューに対してサーバを始動するステップとを含む、方法。

【請求項 12】前記決定するステップが周期的インターバルで実行される、請求項 11 記載の方法。

【請求項 13】異なるサービス・クラスに割当てられる作業要求が、異なるキューに配置され、前記決定するステップが前記キューの各々に対して実行される、請求項 11 記載の方法。

【請求項 14】前記決定するステップが、前記クラスタ内の各システムにより実行される、請求項 11 記載の方法。

【請求項 15】前記決定するステップが、前記キューが前記システム上にサーバを有するか否かを決定するステップと、

前記キューが前記システム上にサーバを有さない場合、前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在するか否かを決定するステップとを含む、請求項 14 記載の方法。

【請求項 16】マシンにより読出され、前記マシンにより実行されて、請求項 11 記載の方法を実行する命令のプログラムを実現する、プログラム記憶装置。

【請求項 17】情報処理システムのクラスタ内において、キュー内の各作業要求を処理可能なサーバの可用性を保証する装置であって、入来作業要求が前記キューに配置され、前記システム上の 1 つ以上のサーバにより処理されるものにおいて、

前記キューに対してサーバを有さない前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在するか否かを決定する手段と、

前記キューに対してサーバを有さない前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在すると決定された場合、前記作業要求が親和性を有する前記サブセット内のシステム上で、前記キューに対してサーバを始動する手段とを含む、装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、第 1 のサービス・クラスに属する入来作業要求が、1 つ以上のサーバによる処理のために、キューに配置される情報処理システムにおいて、サーバの数を制御する方法及び装置に関する。

【0002】

【従来の技術】使用可能なサーバへの割当てのために、入来作業要求がキューに配置されるシステムは周知である。入来作業要求が到来する頻度は、容易に制御されないで、こうしたキュー待機型システムにおいて、システム性能（キュー遅延などにより測定される）を制御する基本手段は、サーバの数を制御することである。従って、サービスされるキューの長さが特定の高いしきい値に達するとき、追加のサーバを始動したり、サービスされるキューの長さが特定の低いしきい値に達するとき、サーバを停止することが知られている。こうした手段は、その設計目的を達成するが、キューに待機された作業要求に加え、他の作業単位がシステム資源を競合するシステムでは不十分である。従って、たとえキューに対して追加のサーバを提供することで、そのキュー内における作業要求の性能を向上しても、こうしたサーバの提供は、システムにより処理される他の作業単位の性能を低下し得、システムの性能を全体として悪化し得る。

【0003】ほとんどの今日のオペレーティング・システム・ソフトウェアは、作業要求に対して指定されるエンドユーザ指向の目的に従い、サーバの数を管理し、同一のコンピュータ・システム内で実行される、独立の目標を有する他の作業を考慮する責任を負うことができない。

い。

【0004】本願の出願人に権利譲渡された 1997 年 3 月 28 日付けの J. D. Aman らによる継続中の米国特許出願第 828440 号は、特定のシステムにおいて、サーバの数を制御する方法及び装置を開示し、そこでは第 1 のサービス・クラスに属する入来作業要求が、1 つ以上のサーバによる処理のためにキューに配置される。本システムは更に、システム資源のドナー (donor) として作用する、少なくとも 1 つの他のサービス・クラスに割当てられる作業単位を有する。本発明によれば、性能測定が第 1 のサービス・クラスの他に、少なくとも 1 つの他のサービス・クラスに対して定義される。サーバを第 1 のサービス・クラスに追加する以前に、第 1 のサービス・クラスの性能測定に及ぼすプラスの効果だけでなく、他のサービス・クラスの性能測定に及ぼすマイナスの効果も決定される。第 1 のサービス・クラスの性能測定に及ぼすプラスの効果は、他のサービス・クラスの性能測定に及ぼすマイナスの効果に勝る場合に限り、サーバが第 1 のサービス・クラスに追加される。

【0005】

【発明が解決しようとする課題】この継続中の特許出願で開示される発明は、サーバを追加するか否かを決定するとき、他の作業に対する影響を考慮するが、これは単一システムの状況においてそのようにする。しかしながら、多重システム・コンプレックス（“シスプレックス (sysplex)”）では、作業要求の 1 つのキューが、コンプレックスに渡り複数のサーバによりサービスされる。従って、所与のキューに対して、サーバを追加するか否かだけでなく、シスプレックス全体の性能を最適化するために、サーバをどこに追加すべきかが決定され得る。

【0006】

【課題を解決するための手段】本発明は、情報処理システムのクラスタ内のサーバの数を制御する方法及び装置に関して、そこでは第 1 のサービス・クラスに属する入来作業要求が、1 つ以上のサーバによる処理のためにキューに配置される。入来作業要求のあるものは、クラスタ内のサーバのサブセット上でのみ実行されるための要求を有し得る。こうした要求を有する作業要求は、それらが実行されなければならないクラスタ内のシステムのサブセットに対して、“親和性 (affinity)” を有するとされる。本発明によれば、サーバがクラスタ内の 1 つ以上のシステム上で始動され、キュー内の作業要求を処理する。これらのサーバを始動するシステムは、システムのクラスタの全容量を利用するように、及び作業要求の親和性要求に応じるように、更にクラスタ内のシステム上で実行されているかも知れない他の作業への影響を最小化するように選択される。新たなサーバが始動されるシステムはまた、システム資源のドナーとして作用する第 2 のサービス・クラスに割当てられる作業単位を有する。本発明によれば、性能測定が第 1 のサービス・ク

ラス同様、第 2 のサービス・クラスに対しても定義される。サーバを第 1 のサービス・クラスに追加する以前に、第 1 のサービス・クラスの性能測定に及ぼすプラスの効果だけでなく、第 2 のサービス・クラスの性能測定に及ぼすマイナスの効果も決定される。第 1 のサービス・クラスの性能測定に及ぼすプラスの効果が、第 2 のサービス・クラスの性能測定に及ぼすマイナスの効果に勝る場合に限り、サーバが第 1 のサービス・クラスに追加される。

【0007】本発明は、複数のユーザ性能目標クラスの各々に対して、各目標クラスの性能目標にもとづき、システムのクラスに渡るサーバの数のシステム管理を可能にする。競合する目標クラスに対して、サーバの追加または除去の影響を考慮するトレードオフが提供される。

【0008】

【発明の実施の形態】本発明を組み込むシステムについて述べる前準備として、作業負荷管理（本発明がそれ上で構成される）の概念に関して前置きすることが適切であろう。

【0009】作業負荷管理は、オペレーティング・システムにより管理される作業単位（プロセス、スレッドなど）が、クラス（サービス・クラスまたは目標クラスと呼ばれる）に編成される概念であり、クラスは予め定義された目標にどれ程よく合致するかに従い、システム資源を提供される。ドナー・クラスからレシーバ（receiver）・クラスへの資源の再割当ては、こうした再割当てから得られるレシーバ・クラスの性能の改善が、ドナー・クラスの性能の劣化に勝る場合、すなわち、予め定義された性能基準により決定される性能における、正味のプラス効果が存在する場合に実行される。このタイプの作業負荷管理は、資源の割当てが資源が再割当てされる作業単位への効果だけにより決定されるのではなく、資源が奪われる作業単位への効果によっても決定されるという点で、大部分のオペレーティング・システムにより実行される“普通の（run-of-the-mill）”資源管理とは異なる。

【0010】この一般的なタイプの作業負荷マネージャは、以下に挙げる特許、特許出願、及び特許以外の刊行物において開示されている。

【0011】D. F. Fergusonらによる米国特許出願第 5 5 0 4 8 9 4 号“Workload Manager for Achieving Transaction Class Response Time Goals in a Multiprocessing System”。

J. D. Amanらによる米国特許第 5 4 7 3 7 7 3 号“Apparatus and Method for Managing a Data Processing System Workload According to Two or More Distinct Processing Goals”。

G. K. Eilertらによる米国特許第 5 5 3 7 5 4 2 号“Apparatus and Method for Managing a Server Workload A

ccording to Client Performance Goals in a Client/Server Data Processing System”。

J. D. Amanらによる米国特許第 5 6 0 3 0 2 9 号“System of Assigning WorkRequests Based on Classifying into an Eligible Class Where the Criteria Goal Oriented and Capacity Information is Available”。

G. K. Eilertらによる米国特許第 5 6 7 5 7 3 9 号“Apparatus and Method for Managing a Distributed Data Processing System Workload According to a Plurality of Distinct Processing Goal Types”。

1995 年 2 月 3 日付けの G. K. Eilertらによる米国特許出願第 3 8 3 0 4 2 号“Multi-System Resource Capping”。

1995 年 6 月 7 日付けの J. D. Amanらによる米国特許出願第 4 8 8 3 7 4 号“Apparatus and Accompanying Method for Assigning Session Requests in a Multi-Server Sysplex Environment”。

1997 年 3 月 2 8 日付けの J. D. Amanらによる米国特許出願第 8 2 8 4 4 0 号“Method and Apparatus for Controlling the Number of Servers in a Client/Server System”。

MVS Planning: Workload Management、IBM 刊行物 GC28-1761-00、1996 年。

MVS Programming: Workload Management Services、IBM 刊行物 GC28-1773-00、1996 年。

【0012】前記特許及び刊行物の中で、米国特許第 5 5 0 4 8 9 4 号及び同第 5 4 7 3 7 7 3 号は、基本的な作業負荷管理システムを開示し、米国特許第 5 5 3 7 5 4 2 号は、米国特許第 5 4 7 3 7 7 3 号の作業負荷管理システムの、クライアント／サーバ・システムへの特定のアプリケーションを開示し、米国特許第 5 6 7 5 7 3 9 号及び米国特許出願第 3 8 3 0 4 2 号は、米国特許第 5 4 7 3 7 7 3 号の作業負荷管理システムの、複数の相互接続システムへの特定のアプリケーションを開示し、米国特許第 5 6 0 3 0 2 9 号は、多重システム・コンプレックス（“シスプレックス”）における作業要求の割当てに関して、米国特許出願第 4 8 8 3 7 4 号は、こうしたコンプレックスにおけるセッション要求の割当てに関して、前述のように、米国特許出願第 8 2 8 4 4 0 号は、多重システム・コンプレックスの 1 つのシステム上でのサーバの数の制御に関する。特許以外の 2 つの刊行物は、IBM（登録商標）OS/390（商標）（以前は MVS（登録商標））オペレーティング・システムにおける作業負荷管理の実現について述べている。

【0013】図 1 は、本発明の一般的な実施例における主要な環境及びフィーチャを示し、相互接続され、協働するコンピュータ・システム 100 のクラス 90 を含み、それらの一般的な 2 つが示されている。本発明の環境は、作業要求 162 のキュー 161 と、クラス 90 に渡り分散され、作業要求をサービスするサーバ 163

のプールとを含む。本発明は、キューに待機された作業の性能目標クラスと、コンピュータ・システム100内で競合する作業の性能目標クラスとにもとづき、サーバ163の数の管理を可能にする。コンピュータ・システム100のクラスタ90に対して1つのポリシーを有することは、分散された作業負荷の単一イメージ・ビューを提供するのに役立つ。当業者であれば、任意の数のコンピュータ・システム100、及び1つのコンピュータ・システム100内の任意の数のこうしたキュー161及びサーバ163のグループが、本発明の趣旨及び範囲から逸れることなく使用され得ることが理解されよう。

【0014】コンピュータ・システム100は分散された作業負荷を実行し、各コンピュータ・システムが、例えばIBM OS/390オペレーティング・システムなどの、オペレーティング・システム101の自身のコピーにより制御される。

【0015】それぞれのコンピュータ・システム100上のオペレーティング・システム101の各コピーは、本願で述べられるステップを実行する。説明の中で“ローカル”・システム100を指し示すとき、それは述べられているステップを実行しているシステム100を意味する。一方、“リモート”・システム100は、管理される他の全てのシステム100を意味する。各システム100がそれ自身を局所的（ローカル）と見なし、他の全てのシステム100を遠隔的と見なす点に留意されたい。

【0016】本発明に関する改良以外は、システム100は、継続中の米国特許出願第828440号及び米国特許第5675739号で開示されるシステムと類似である。図1に示されるように、システム100は複数の相互接続されるシステム100の1つであり、これらは同様に管理され、クラスタ90（システム・コンプレックスまたはシブプレックスとも呼ばれる）を構成する。米国特許第5675739号で教示されるように、作業単位が分類される様々なサービス・クラスの性能が、特定のシステム100に対してだけでなく、クラスタ90全体に対しても追跡され得る。この目的のために、また後述の説明から明らかになるように、システム100とクラスタ90内の他のシステム100との間で、性能結果を伝達し合う手段が提供される。

【0017】ディスパッチャ102は、オペレーティング・システム101の構成要素であり、次にコンピュータにより実行される作業単位を選択する。作業単位は、コンピュータ・システム100の目的に相当する有用な作業を実行するアプリケーション・プログラムである。実行準備が整った作業単位は、アドレス空間制御ブロック（ASCB）・キューと呼ばれる、オペレーティング・システム・メモリ内の制御ブロックの連鎖により表される。

【0018】作業マネージャ160は、オペレーティン

グ・システム101の外部の構成要素であり、これはオペレーティング・システム・サービスを用いて、1つ以上のキュー161を作業負荷マネージャ（WLM）105に定義し、作業要求162をこれらのキュー上に挿入する。作業マネージャ160は、挿入された作業要求162を、クラスタ90内の任意のシステム100上の作業マネージャ160のサーバ163による選択のために、先入れ先出し（FIFO）順に保持する。作業マネージャ160は、サーバが、それが実行されているシステム100と親和性を有する要求だけを選択することを保証する。

【0019】サーバ163は作業マネージャ160の構成要素であり、キューに待機された作業要求162をサービスすることができる。作業負荷マネージャ105がサーバ163を始動し、作業マネージャ160のキュー161上の作業要求162をサービスするとき、作業負荷マネージャ105は、共用データ機構140上に記憶されたサーバ定義を用いて、アドレス空間（すなわちプロセス）164を始動する。作業負荷マネージャ105により始動されるアドレス空間164は、作業負荷マネージャ105の指示に従い、そのアドレス空間がサービスすべき特定のキュー161上の要求162をサービスする、1つ以上のサーバ（すなわちディスパッチ可能単位またはタスク）163を含む。

【0020】図2は、コンピュータ・システム100が接続されるネットワーク（図示せず）から、作業負荷マネージャ105により管理されるサーバ・アドレス空間164への、クライアント作業要求162の流れを示す。作業要求162はクラスタ90内の特定のコンピュータ・システム100に経路指定され、作業マネージャ160により受信される。作業要求162の受信に際して、作業マネージャ160はそれを作業負荷マネージャ（WLM）・サービス・クラスに分類し、作業要求を作業キュー161に挿入する。作業キュー161は、クラスタ90内の全てのコンピュータ・システム100により共用される。すなわち、作業キュー161はクラスタ全体に渡るキューである。作業要求162は、それを実行する準備が整ったサーバが現れるまで、作業キュー161内で待機する。

【0021】新たな作業要求162を実行する準備が整った（アドレス空間が丁度始動されたか、タスクが前の要求の実行を終了したとき）、クラスタ90内のあるシステム100上のサーバ・アドレス空間164内のタスク163が、新たな作業要求のために作業マネージャ160を呼び出す。アドレス空間164がサービスしている作業キュー161上に作業要求162が存在し、その作業要求がサーバが実行されているシステム100と親和性を有する場合、作業マネージャ160はその作業要求をサーバ163に受け渡す。それ以外では、作業マネージャ160は作業要求162が有効になるまで、サー

バ 1 6 3 を延期する。

【0022】作業要求 1 6 2 が作業マネージャ 1 6 0 により受信されるとき、それは作業キュー 1 6 1 上に配置され、サーバ 1 6 3 が作業要求を実行するために使用可能になるのを待機する。作業要求 1 6 2 の作業マネージャ 1 6 0、アプリケーション環境名、及び WLM サービス・クラスのそれぞれの固有の組み合わせに対して、1 つの作業キュー 1 6 1 が存在する。(アプリケーション環境は、類似のクライアント作業要求 1 6 2 のセットが実行されるべき環境である。OS/390 条件では、これは作業要求を実行するために、サーバ・アドレス空間を始動するために使用されるジョブ制御言語 (JCL) プログラムにマップされる。) 特定の作業キュー 1 6 1 に対して、第 1 の作業要求 1 6 2 が到来するとき、キューイング構造が動的に生成される。作業キュー 1 6 1 に対して、所定時間 (例えば 1 時間) 何も活動が発生しなかった場合、構造は消去される。新たな WLM ポリシを活動化するなどのように、キューに待機された作業要求 1 6 2 の WLM サービス・クラスを変更し得るアクションが発生する場合、作業負荷マネージャ 1 0 5 は作業マネージャ 1 6 0 にその変更を知らせ、作業マネージャ 1 6 0 は、各作業要求 1 6 2 の新たな WLM サービス・クラスを反映するように、作業キュー 1 6 1 を再生する。

【0023】サーバ 1 6 3 を有さないシステム 1 0 0 と親和性を有する作業要求 1 6 2 は、他のシステム 1 0 0 上に、その作業要求のサービス・クラスの目標を満たす上で十分なサーバが存在する場合、決して実行されないといった危険性が存在する。この危険性を回避するために、作業負荷マネージャ 1 0 5 は、クラスタ 9 0 内のどこかに、キュー 1 6 1 上の各作業要求を実行可能な少なくとも 1 つのサーバ 1 6 3 が存在するように保証する。図 8 はこの論理を示す。この論理は、クラスタ 9 0 内の各コンピュータ・システム 1 0 0 上の作業マネージャ 1 6 0 により実行される。

【0024】ステップ 7 0 1 で、作業マネージャ 1 6 0 が自身が所有する第 1 のキュー 1 6 1 を調査する。ステップ 7 0 2 で、作業マネージャ 1 6 0 が、ローカル・システム 1 0 0 上にこのキュー 1 6 1 に対するサーバ 1 6 3 が存在するか否かをチェックする。サーバ 1 6 3 が存在する場合、作業マネージャ 1 6 0 は次のキュー 1 6 1 に移行する (ステップ 7 0 8 乃至 7 0 9)。作業マネージャ 1 6 0 が、局所的にサーバ 1 6 3 を有さないキュー 1 6 1 を見出す場合、作業マネージャ 1 6 0 は次にキュー上の各作業要求を調査し、第 1 の作業要求を開始する (ステップ 7 0 3 乃至 7 0 6)。各作業要求 1 6 2 に対して、作業マネージャ 1 6 0 は、クラスタ 9 0 内のどこかに、現作業要求を実行可能なサーバ 1 6 3 が存在するか否かをチェックする (ステップ 7 0 4)。現作業要求を実行可能なサーバ 1 6 3 が存在する場合、作業マネ

ージャ 1 6 0 はキュー 1 6 1 上の次の作業要求 1 6 2 に移行する (ステップ 7 0 6)。現作業要求を実行可能なサーバ 1 6 3 が存在しない場合、作業マネージャ 1 6 0 は作業負荷マネージャ 1 0 5 を呼び出し、サーバ 1 6 3 を始動し (ステップ 7 0 7)、次のキュー 1 6 1 に移行する (ステップ 7 0 8 乃至 7 0 9)。作業マネージャ 1 6 0 は、作業マネージャにより所有される全てのキュー 1 6 1 が処理されるまで (ステップ 7 1 0)、同様に継続する。

【0025】作業マネージャ 1 6 0 が作業負荷マネージャ 1 0 5 を呼び出すとき (ステップ 7 0 7)、作業要求に対してサーバを始動すべき最善のシステム 1 0 0 を決定するために、作業負荷マネージャ 1 0 5 は、クラスタ 9 0 内の各システム 1 0 0 に対して、各重要度にて有効なサービス、及びそのシステムに対して未使用のサービスを示す、サービス有効配列 (Service Available Array) を保持する。この配列は下記のように、各重要度に対するエントリ、及び未使用のサービスに対するエントリを含む。

【0026】

配列要素	配列要素内容
配列要素 1	重要度 0 にて有効なサービス
配列要素 2	重要度 1 にて有効なサービス
配列要素 3	重要度 2 にて有効なサービス
配列要素 4	重要度 3 にて有効なサービス
配列要素 5	重要度 4 にて有効なサービス
配列要素 6	重要度 5 にて有効なサービス
配列要素 7	重要度 6 にて有効なサービス
配列要素 8	未使用サービス

【0027】サービス有効配列は、本願の出願人に権利譲渡された、1997 年 3 月 28 日付けの G. K. Eilert らによる米国特許出願第 8 2 7 5 2 9 号 "Managing Processor Resources in a Multisystem Environment" でも述べられている。

【0028】作業負荷マネージャ 1 0 5 は、要求のサービス・クラスの重要度にて有効な最大限のサービスにより、システム 1 0 0 上で新たなサーバを始動する。続くアドレス空間 1 6 4 は、作業負荷をサポートすることが要求されるときに始動される (後述のポリシ調整を参照)。好適には、アドレス空間 1 6 4 を始動する機構は、自動的にアドレス空間を始動する他の実施例における共通の問題を回避するための、幾つかのフィーチャを有する。従って、アドレス空間 1 6 4 の始動は好適には、1 度に 1 つの始動だけが進行するように歩調を合わされる。この歩調合わせは、システム 1 0 0 にアドレス空間 1 6 4 の始動が殺到することを回避する。

【0029】また、所定の連続的な始動失敗 (例えば 3 度の失敗) に遭遇するとき、所与のアプリケーション環境においては、追加のアドレス空間 1 6 4 の生成を回避するための特殊論理が、好適には提供される。こうした

失敗が起こり得る原因として、アプリケーション環境における、JCLプロシージャのJCLEエラーが考えられる。前述の特殊論理の提供は、JCLEエラーが訂正されるまで、成功裡に始動しないアドレス空間を始動しようとするループに入り込むことを回避する。

【0030】更に、作業要求162を実行する間に、サーバ・アドレス空間164が失敗する場合、作業負荷マネージャ105は好適には、それを置換するための新たなアドレス空間を始動する。失敗が繰り返されると、作業負荷マネージャ105は、オペレータ・コマンドにより問題が解決されたことを知らされるまで、そのアプリケーション環境において、作業要求の受諾を停止する。

【0031】所与のサーバ・アドレス空間164は、たとえそれが通常、1つの作業キュー161だけをサービスするとしても、そのアプリケーション環境において、物理的にあらゆる作業要求162をサービスすることができる。好適には、サーバ・アドレス空間164は、もはやその作業キュー161をサポートする必要がないときでも、即時終了されない。代わりに、サーバ・アドレス空間164は、ある期間“フリー・エージェント”として待機し、同一のアプリケーション環境において、別の作業キュー161をサポートするために使用され得るか否かを調査する。サーバ・アドレス空間164が新たな作業キュー161にシフトされ得る場合、その作業キューに対して新たなサーバ・アドレス空間を始動するサーバヘッドが回避される。所定時間（例えば5分）内に、サーバ・アドレス空間164が別の作業キュー161により必要とされない場合、それは終了される。

【0032】本発明は入力としてシステム管理者により確立され、データ記憶機構140に記憶された性能目標141及びサーバ定義を受け取る。データ記憶機構140は、管理される各システム100によりアクセス可能である。ここで示される性能目標には2つのタイプ、すなわち応答時間（秒）と、実行速度（%）とがある。当業者であれば、本発明の趣旨及び範囲から逸れることなく、他の目標または追加の目標が選択され得ることが理解できよう。性能目標には、各目標の相対重要度の指定が含まれる。性能目標141が、管理される各システム100のオペレーティング・システム101の作業負荷マネージャ要素105により、システム内に読出される。システム管理者により確立され、指定された性能目標の各々は、各システム100上の作業負荷マネージャ105に、個々の作業単位が割当てられる性能クラスを確立させる。各性能クラスがクラス・テーブル・エントリ106により、オペレーティング・システム101のメモリ内に表される。（内部表現にて）指定された目標、及び性能クラスに関する他の情報が、クラス・テーブル・エントリに記録される。クラス・テーブル・エントリに記憶される他の情報には、サーバ163の数107（制御変数）、目標クラスの相対重要度108（入力

値）、多重システム性能指標（PI）151、ローカル性能指標152（性能測定を表す計算値）、応答時間目標110（入力値）、実行速度目標111（入力値）、サンプル・データ113（測定データ）、リモート応答時間履歴（157）（測定データ）、リモート速度履歴158（測定データ）、サンプル・データ履歴125（測定データ）、及び応答時間履歴126（測定データ）が含まれる。

【0033】オペレーティング・システム101はシステム資源マネージャ（SRM）112を含み、これは多重システム目標駆動型制御装置（MGDPC）114を含む。これらの構成要素は一般に、米国特許第5473773号及び同第5675739号で述べられるように動作する。しかしながら、MGDPC114は本発明に従い、サーバ163の数を管理するように変更される。MGDPC114は後述のように、目標の達成度を測定する機能、改善された性能を必要とするユーザ性能目標クラスを選択する機能、及び、関連作業単位の制御変数を変更することにより、選択されたユーザ性能目標クラスの性能を改善する機能を実行する。好適な実施例では、MGDPC機能はおよそ毎10秒ごとの周期的なタイマ満了にもとづき、周期的に実行される。MGDPC機能が実行されるインタバルは、MGDPCインタバルまたはポリシ調整インタバルと呼ばれる。

【0034】MGDPC114の動作の一般的な態様は、米国特許第5675739号で述べられるように、次のようである。ブロック115で、各ユーザ性能目標クラス106に対して、指定目標110または111を用いて、多重システム性能指標151及びローカル性能指標152が計算される。多重システム性能指標151は、管理される全てのシステム100に渡る目標クラスに関連付けられる作業単位の性能を表す。ローカル性能指標152は、ローカル・システム100上の目標クラスに関連付けられる作業単位の性能を表す。結果の性能指標151、152は、対応するクラス・テーブル・エントリ106に記録される。ユーザ性能目標の達成度を測定する方法としての性能指標の概念は周知である。例えば、前記のFergusonらによる米国特許第5504894号では、性能指標として、実際の応答時間が目標応答時間により除算されるように述べられている。

【0035】ブロック116で、ユーザ性能目標クラスが選択され、相対目標重要度108及び性能指標151、152の現在値の順で、性能改善度を受信する。選択されたユーザ性能目標クラスは、レシーバ（receiver）と呼ばれる。MGDPC114はレシーバを選択するとき、最初に多重システム性能指標151を用いることにより、管理される全てのシステム100に渡り作業単位が目標を満足するために、最も大きな影響を有するアクションを実行する。多重システム性能指標151にもとづき実行すべきアクションが存在しない場合、ロー

カル性能指標 152 が用いられ、ローカル・システム 100 がその目標を満足するために最も有用なレシーバが選択される。

【0036】候補のレシーバ・クラスが決定された後、周知のように、状態サンプル 125 を用いて性能ボトルネックを構成する、そのクラスの制御変数がブロック 117 で決定される。米国特許第 5675739 号で述べられるように、制御変数には、保護プロセッサ記憶ターゲット（ページング遅延に影響）、スワップ保護時間（SPT）ターゲット（スワップ遅延に影響）、多重プログラミング・レベル（MPL）・ターゲット（MPL 遅延に影響）、及びディスパッチ優先順位（CPU 遅延に影響）などの変数が含まれる。本発明によれば、制御変数として、キュー遅延に影響を及ぼすサーバ 163 の数が含まれる。

【0037】図 1 では、サーバ 163 の数 107 が、クラス・テーブル・エントリ 106 に記憶されて示され、これは 1 クラス当たり、1 つのキュー 161 が限度であることを意味する。しかしながら、これは単に説明の簡略化のためであり、当業者であれば、単にデータの位置を変更することにより、1 クラス当たり複数のキュー 161 が独立に管理され得ることが理解できよう。基本的な必要条件是、1 つのキュー 161 に対する作業要求 162 が 1 つの目標だけを有すること、各サーバ 163 が要求をサービスするための等しい能力を有すること、及び作業負荷マネージャ 105 からの（への）通知無しでは、サーバが 2 つ以上のキュー 161 上の作業をサービスすることができないことである。

【0038】候補の性能ボトルネックが識別された後、制御変数の潜在的な変化がブロック 118 で考慮される。ブロック 123 で、相対目標重要度 108 及び性能指標 151、152 の現在値にもとづき、性能低下が起こり得るユーザ性能目標クラスが選択される。従って、選択されたユーザ性能目標クラスはドナーと呼ばれる。

【0039】候補ドナー・クラスが選択された後、提案された変化がブロック 124 で評価される。具体的には、サーバ 163 の数 107、及び前述され、また米国特許第 5675739 号でも述べられる変数を含む制御変数の各々に対して、レシーバ及びドナーの両者において、多重システム性能指標 151 及びローカル性能指標 152 の期待変化に対するネット値が評価される。提案された変化は、結果が目標に対して、ドナーに与える損害よりも多くの改善をレシーバにもたらす場合、ネット値を有する。提案された変化がネット値を有する場合、それぞれの制御変数がドナー及びレシーバの両者に対して調整される。

【0040】管理される各システム 100 はデータ伝送機構 155 に接続され、これは各システム 100 がデータ・レコードをあらゆる他のシステム 100 に送信することを可能にする。ブロック 153 で、各目標クラスの

最近の性能を記述するデータ・レコードが、あらゆる他のシステム 100 に送信される。

【0041】多重システム目標駆動型性能制御装置（MGDP C）機能は、周期的に実行され（好適な実施例では毎 10 秒ごとに 1 度）、タイマ満了を介して呼び出される。MGDP C の機能は、性能問題の増分検出及び訂正のためのフィードバック・ループを提供し、オペレーティング・システム 101 を適応され、自己同調させる。

【0042】米国特許第 5675739 号で述べられるように、MGDP C インタバルの終わりに、インタバルの間の各目標クラスの性能を記述するデータ・レコードが、管理される各システム 100 に送信される。応答時間目標を有する性能目標クラスに対して、このデータ・レコードは目標クラス名と、リモート応答時間履歴の行（row）に等価なエントリを有する配列とを含む。ここでリモート応答時間履歴は、最後の MGDP C インタバルに渡る目標クラスの完了を記述する。速度目標を有する目標クラスに対して、このデータ・レコードは目標クラス名と、目標クラス内の作業が最後の MGDP C インタバル内で実行中にサンプリングされた回数と、目標クラス内の作業が最後の MGDP C インタバル内で、実行中または遅延中にサンプリングされた回数とを含む。本発明によれば、各システム 100 が追加のデータとして、データを送信するシステム 100 のサービス有効配列、各キュー 161 に対するサーバ 163 の数、及び各キュー 161 に対する遊休サーバ 163 の数を送信する。

【0043】ブロック 154 で、リモート・データ・レシーバが MGDP C 114 とは非同期に、リモート・システム 100 から性能データを受信する。受信データは MGDP C 114 による後の処理のために、リモート性能データ履歴（157、158）に配置される。

【0044】図 3 は、解決する資源ボトルネックを選択するために、ボトルネック発見手段 117 により使用される状態データを示す。各遅延タイプに対して、性能目標クラス・テーブル・エントリ 106 が、その遅延タイプに遭遇するサンプルの数、及び MGDP C 114 の現呼び出しの間に、その遅延タイプが既にボトルネックとして選択されたか否かを示すフラグを含む。クロス・メモリ・ページング・タイプ遅延の場合、クラス・テーブル・エントリ 106 が、遅延に遭遇したアドレス空間の識別子を含む。

【0045】ボトルネック発見手段 117 の論理フローが図 4 に示される。解決するボトルネックの選択は、MGDP C 114 の現呼び出しの間にまだ選択されておらず、最も多くのサンプルを有する遅延タイプを選択することにより実行される。遅延タイプが選択されると、フラグがセットされ、それによりボトルネック発見手段 117 が、MGDP C 114 のこの呼び出しの間に再度呼

び出される場合、その遅延タイプがスキップされる。

【0046】図4のステップ501では、CPU遅延タイプが、まだ選択されていない全ての遅延タイプの中で、最も多くの遅延サンプルを有するか否かが判断される。肯定の場合、ステップ502で、CPU遅延選択フラグがセットされ、CPU遅延が次に解決されるべきボトルネックとして返却される。

【0047】ステップ503では、MPL遅延タイプが、まだ選択されていない全ての遅延タイプの中で、最も多くの遅延サンプルを有するか否かが判断される。肯定の場合、ステップ504で、MPL遅延選択フラグがセットされ、MPL遅延が次に解決されるべきボトルネックとして返却される。

【0048】ステップ505では、スワップ遅延タイプが、まだ選択されていない全ての遅延タイプの中で、最も多くの遅延サンプルを有するか否かが判断される。肯定の場合、ステップ506で、スワップ遅延選択フラグがセットされ、スワップ遅延が次に解決されるべきボトルネックとして返却される。

【0049】ステップ507では、ページング遅延タイプが、まだ選択されていない全ての遅延タイプの中で、最も多くの遅延サンプルを有するか否かが判断される。肯定の場合、ステップ508で、ページング遅延選択フラグがセットされ、ページング遅延が次に解決されるべきボトルネックとして返却される。5つのタイプのページング遅延が存在する。ステップ507では、最も多くの遅延サンプルを有するタイプが突き止められ、ステップ508で、特定のタイプに対してフラグがセットされ、その特定のタイプが返却される。好適な実施例の環境(OS/390)において周知のように、ページング遅延のタイプには、専用領域、共通領域、クロス・メモリ、仮想入出力(VIO)、及びハイパ空間が存在し、各々がページング遅延状況に対応する。

【0050】最後にステップ509で、キュー遅延タイプが、まだ選択されていない全ての遅延タイプの中で、最も多くの遅延サンプルを有するか否かが判断される。クラスはキュー161上の各作業要求に対して、ローカル・システム100上で実行される資格のある1つのキュー遅延タイプ・サンプルを獲得する。肯定の場合、ステップ510で、キュー遅延選択フラグがセットされ、キュー遅延が次に解決されるべきボトルネックとして返却される。キュー遅延は、クラスタ90内の別のシステム100が最後のポリシ調整インタバルの間に、キュー161に対してサーバ163を始動した場合、ローカル・システム100上では解決されない。キュー遅延はまた、候補レシーバ・クラスが準備完了の作業をスワップ・アウトした場合にも、解決されない。

【0051】次のセクションでは、ボトルネック発見手段117により選択された遅延を低減するように、制御変数を変更することにより、如何にレシーバ性能目標ク

ラス性能が改善されるか、そして特に、レシーバにより遭遇されるキュー遅延を低減することにより、如何に性能が改善されるかについて述べる。共用キュー161の場合、これは2ステップ・プロセスである。第1に、ローカル・システム100上にサーバ163を追加することにより、評価が行われる。この評価には、ドナー作業に対する影響が含まれる。サーバ163の追加において、ネット値が存在する場合、次のステップでサーバがローカル・システム100上で始動されるべきか、或いはそれらがクラスタ90内の別のシステム100上で始動されるべきかが決定される。リモート・システム100がサーバ163を始動するために、より好適であると思われる場合、ローカル・システム100はそのリモート・システムにサーバを始動する機会を与えるために待機する。しかしながら、そのリモート・システム100がサーバ163を始動しない場合、ローカル・システム100が図9に関連して後述されるように、それらを始動する。

【0052】図5は、追加のサーバ163を始動することによる、性能の改善を評価する論理フローを示す。図5乃至図7は、固定手段118によりネット値手段124に提供される性能指標デルタ予測を生成するステップを示す。ステップ1401で、サーバ163の新たな数評価のために選択される。この数は変化を価値あるものにするために、十分なレシーバ値(ステップ1405でチェックされる)を生成するように十分大きくなければならない。一方、この数は、追加のサーバ163の値に限られるように、例えばキューに待機されて実行される作業要求の総数より大きくならないように、余り大きくてはならない。次のステップは、追加のサーバ163が使用する追加のCPUを計算する。これは作業要求により使用される平均CPUに、追加されるサーバ163を乗算することにより実行される。

【0053】ステップ1402で、新たな数のサーバ163における作業要求162の予測数が、図6に示されるサーバ準備完了ユーザ平均グラフから読出される。ステップ1403で、現予測キュー遅延が、図7に示されるキュー遅延グラフから読出される。ステップ1404で、ローカル性能指標デルタ及び多重システム性能指標デルタの予測値が計算される。これらの計算は以下で示される。

【0054】図6は、キュー準備完了ユーザ平均グラフを示す。キュー準備完了ユーザ平均グラフは、キュー161に対するサーバ163の数の変化を評価するとき、サーバ163に対する需要を予測するために使用される。このグラフは、作業要求162がバックアップを開始するポイントを示す。横座標(x)値は、キュー161が使用可能なサーバ163の数である。縦座標(y)値は、実行準備完了の作業要求162の最大数である。

【0055】図7は、キュー遅延グラフを表す。キュー遅延グラフは、キュー161に対するサーバ163の数を増減する値を評価するために使用される。このグラフは、キュー・サーバ163の数を増加することにより、如何に応答時間が改善されるか、或いはキュー・サーバ163の数を減少することにより、如何に応答時間が低下されるかを示す。グラフはまた、例えばデータベース・ロック競合などの、追加のサーバ163を追加することにより生じ得る、作業負荷マネージャ105により管理されない資源に対する競合を暗黙的に考慮する。こうした場合では、追加のサーバ163が追加されても、グラフ上のキュー遅延が減少しない。横座標値は、使用可能なサーバ163を有し、システム100のクラスタ90に渡りスワップ・インされる準備完了の作業要求162の割合である。縦座標値は1完了当たりのキュー遅延である。

【0056】サーバ163の数の増加に対する、シスプレックス（すなわち多重システム）性能指標（PI）デルタは、次のように計算される。ここでシスプレックス性能指標デルタが計算される理由は、キュー161がシスプレックス全体に渡る資源であるからである。

【0057】応答時間目標に対して、

$(\text{予測シスプレックスPIデルタ}) = (\text{予測キュー遅延} - \text{現キュー遅延}) / \text{応答時間目標}$

【0058】速度目標に対して、

$(\text{新シスプレックス速度}) = \{ \text{cpuu} + (\text{cpuu} / \text{oldserver}) * \text{newserver} \} / \{ \text{non-idle} + ((\text{qd} / \text{qreq}) * (\text{oldserver} - \text{newserver})) \}$

$(\text{シスプレックスPIデルタ}) = (\text{現シスプレックスPI} - \text{目標}) / \text{新シスプレックス速度}$

【0059】ここでcpuuは、シスプレックスCPU使用サンプル；oldserverは、評価される変化が実行される前のサーバ163の数；newserverは、評価される変化が実行された後のサーバ163の数；non-idleは、シスプレックス非遊休サンプルの総数；qdは、シスプレックス・キュー遅延サンプル；qreqは、キュー161上の作業要求162の数である。

【0060】同様の計算が、サーバ163の数の減少に対する性能指標デルタを計算するために使用される。

【0061】ステップ1405で、追加のサーバ163の数により提供される十分なレシーバ値に対してチェックが行われる。好適には、このステップは、新たなサーバ163が、それらの追加を価値あるものにするために十分なCPU時間を獲得するかどうかを判断するステップを含む。十分なレシーバ値が存在しない場合、制御はステップ1401に戻り、より多くのサーバ163が評価のために選択される。

【0062】十分なレシーバ値が存在する場合、ステップ1406でドナー選択手段123が呼び出され、レシーバ性能目標クラスのために、追加のサーバ163を始

動するために必要な記憶装置のドナーを見い出す。

【0063】ドナー・クラスのために調整される制御変数は、必ずしもそのクラスに対するサーバ163の数107である必要はない。ドナー・クラスの幾つかの異なる制御変数の任意の1つ、例えばMPLスロットまたは保護プロセッサ記憶装置などが、代わりにまたは追加として調整され、追加のサーバ163のために必要な記憶装置を提供してもよい。本発明の一部を成すものではないが、こうした制御変数の調整がドナー・クラスに与える効果を評価する方法が、米国特許第5537542号及び同第5675739号で述べられている。

【0064】ステップ1407では、ドナーから記憶装置を受け取り、レシーバ・クラスのためにサーバ163の数を増加するに際して、ネット値が存在することを保証するためにチェックが行われる。米国特許第5675739号で述べられるように、これは1つ以上の幾つかの異なる基準、例えば資源再割当ての後に、ドナーがその目標に合致するように予測されるか否か、レシーバが現在その目標を逸しつつあるか否か、レシーバがドナーよりも重要なクラスか否か、またはドナー及びレシーバの結合性能指標において、ネット利得が存在するか否か、すなわち、サーバをレシーバ・クラスに追加することが、レシーバ・クラスの性能指標に及ぼすプラスの効果が、ドナー・クラスの性能指標に及ぼすマイナスの効果を上回るか否かなどの基準を用いて決定される。ネット値が存在する場合、次のステップで、ローカル・システム100が新たなサーバ163を始動するための（ステップ1408）、クラスタ90内の最善のシステムか否かが決定される。ネット値が存在しない場合、レシーバ目標クラス・キュー遅延問題は解決され得ない（ステップ1409）。

【0065】図9は、新たなサーバ163を始動するクラスタ90内の最善のシステム100を表す、ターゲット・システムを決定するプロシージャを示す。このプロシージャは、一旦サーバの追加に対してネット値が存在し、1つ以上のサーバ163がレシーバ・クラスに追加されるべきことが決定されると、図5のステップ1408の一部として、クラスタ90内の各システム100により実行される。ローカル・システム100が最初に、クラスタ90内の任意のシステム100が他の作業に影響すること無く、新たなサーバ163をサポートするための十分な遊休容量を有するか否かをチェックする（ステップ801）。これはクラスタ90内の各システム100のサービス有効配列を調査し、配列要素8において、新たなサーバ163をサポートするために使用可能な十分なCPUサービス（未使用CPUサービス）を有するシステム100を選択することにより実行される。複数のシステム100が十分な未使用CPUサービスを有する場合、最も多くの未使用サービスを有するシステム100が選択される。しかしながら、キューに待機さ

れた作業要求が存在するときに、システム100が遊休状態のサーバを有する場合、これは多くの作業要求がそのシステム100に対して親和性を持たないことを意味するので、この遊休状態のサーバ163を有するシステム100は選択されない。

【0066】ローカル・システム100は次に、サーバ163を始動するための、十分な遊休状態のCPU容量を有するターゲット・システム100が見い出されたか否かをチェックする(ステップ802)。ターゲット・システム100が見い出され、それがローカル・システム100の場合(ステップ803)、ローカル・システム100がサーバ163を局所的に始動する(ステップ804乃至805)。

【0067】ステップ803で、別のシステム100が新たなサーバ163を始動するための最も多くの遊休容量を有することが見い出されると、制御はステップ806に移行し、ローカル・システム100がポリシ調整インタバル(10秒)を待機し、別のシステム100がサーバ163を始動したか否かをチェックする(ステップ807)。別のシステム100がサーバ163を始動した場合、局所的に何もアクションは実行されない(ステップ811)。他のシステム100がサーバ163を始動しなかった場合、ローカル・システム100は、自身が新たなサーバ163をサポートする十分な遊休CPU容量を有するか否かをチェックする(ステップ812)。有する場合、ローカル・システム100は局所的にサーバ163を始動する(ステップ813乃至814)。

【0068】新たなサーバ163をサポートするための十分な遊休CPU容量を有するシステム100が存在しなかった場合、或いは、十分な遊休CPU容量を有するシステム100が存在したが、そのシステム100がサーバを始動しなかった場合、制御はステップ808に移行する。こうしたシステム100がサーバ163を始動しない1つの理由は、メモリが欠乏するためである。この時点で、ドナー作業に影響を及ぼすこと無しに、新たなサーバ163が始動され得ないことが判明する。従って、ローカル・システム100は、局所的なサーバ163の始動により、ドナー作業がその目標を逸するか否かをチェックする。ドナー作業がその目標を逸しない場合、ローカル・システム100はサーバ163を局所的に始動する(ステップ817乃至818)。サーバ163の局所的な始動により、ドナー・クラスがその目標を逸する場合には、ローカル・システム100はドナー作業への影響が最も小さいシステム100を見い出す(ステップ809)。

【0069】図10は、図9のステップ809で、ドナー作業への影響が最も小さいシステム100を決定するルーチンを示す。ルーチンは最初に、ドナー・クラスの名前及びドナーの性能指標(PI)デルタを、クラス

90内の他のシステムに送信する(ステップ901)。このドナー情報を交換することにより、レシーバ・クラスに対するサーバ163の追加を評価している各システム100は、他の全てのシステム100へのサーバ追加の影響を調査する。ルーチンは次に、他のシステム100がそれらのドナー情報を送信することを可能にするために、1ポリシ・インタバル(示される実施例では10秒)を待機する(ステップ902)。ルーチンは次に、ドナー・クラスが最も低い重要度を有するシステム100を選択し(ステップ903)、選択されたシステム100を呼び出しルーチンに返却して、図9のステップ809を完了する(ステップ905)。ドナー重要度に対等なものが存在する場合(ステップ904)、ルーチンはドナーの性能指標(PI)デルタが最小のシステム100を選択し(ステップ906)、このシステム100を呼び出しルーチンに返却する(ステップ907)。

【0070】再度図9を参照して、ステップ809を完了後、ローカル・システム100は、ドナーに対して最も小さな影響を有するとして選択されたシステム100が、ローカル・システムであるか否かをチェックする(ステップ810)。そうである場合、ローカル・システム100がサーバ163を局所的に始動する(ステップ817乃至818)。それ以外では、ローカル・システム100は、別のシステム100がサーバ163を始動することを可能にするために、1ポリシ・インタバル待機し(ステップ815)、このインタバルの終りに別のシステム100がサーバ163を始動したか否かをチェックする(ステップ816)。別のシステム100がサーバ163を始動した場合、ローカル・システム100は何もアクションを起こさない(ステップ818)。他のシステム100がサーバ163を始動しなかった場合、ローカル・システム100はそれらを局所的に始動する(ステップ817乃至818)。

【0071】図5のステップ1408では、特定の環境下のキュー161に対して、新たなサーバ163を始動する要求を一時的に延期する論理が含まれる。既存の作業への不要な影響を回避するために、新たなサーバ163を始動する並列要求が制限される。この歩調合わせは、オペレーティング・システム101に、追加のサーバ163を始動するための多くの並列要求が殺到し、混乱することを回避する。システム管理者により提供される、データ・レポジトリ141内の誤った情報の検出も実行され、新たなサーバ163が成功裡に始動され得ない程、サーバ定義情報が不正確な場合に、無限の再試行ループを阻止する。一旦サーバ163が始動されると、サーバが予期せず故障した場合に、それを自動的に置換する論理も含まれる。同一のサーバ定義情報を有するが、同一の作業マネージャ160に対して異なるキュー161をサービスする遊休サーバ163が、キュー間で移動され得ることにより、特定のキューに対してサーバ

163の数を増加する要求が満足され、完全に新たなサーバを始動するオーバーヘッドを回避する。

【0072】本発明は好適には、1つ以上のハードウェア・マシン上で実行されるソフトウェア（すなわち、プログラム記憶装置上で実現される命令のマシン読出し可能プログラム）として実現される。これまで特定の実施例について述べてきたが、当業者であれば、ここで特定の述べられた以外の他の実施例も、本発明の趣旨から逸れることなく実現され得ることが明らかであろう。また、当業者であれば、様々な等価な要素が、ここで特定の開示された要素の代わりに代用され得ることが理解されよう。同様に、ここで開示された実施例の変更及び組み合わせも明らかであろう。例えば、各サービス・クラスに対して、ここで開示された1つのキューではなく、複数のキューが提供され得る。開示された実施例及びそれらの詳細は、本発明の実施例を教示することを意図したのものであって、本発明を制限するものではない。従って、こうした明白ではあるが、ここで開示されなかった変更及び組み合わせも、本発明の趣旨及び範囲に含まれるものと見なされる。

【0073】まとめとして、本発明の構成に関して以下の事項を開示する。

【0074】(1) 情報処理システムのクラスタ内においてサーバの数を制御する方法であって、あるサービス・クラスに属する入来作業要求が、前記クラスタ全体に渡るキューに配置され、前記クラスタの前記システム上の1つ以上のサーバにより処理されるものにおいて、1つ以上のサーバが前記サービス・クラスに追加されるべきか否かを決定するステップと、1つ以上のサーバが前記サービス・クラスに追加されるべきと決定される場合、前記サーバが追加されるべき、前記クラスタ内のターゲット・システムを決定するステップと、前記サーバを前記ターゲット・システム上に追加するステップとを含む、方法。

(2) 前記決定するステップが前記システムの各々により実行される、前記(1)記載の方法。

(3) 前記システムの各々が、該システムがターゲット・システムか否かを決定し、前記ターゲット・システムであると決定するとき、1つ以上のサーバを局所的に追加する、前記(2)記載の方法。

(4) 前記サービス・クラスが第1のサービス・クラスであり、前記クラスタが少なくとも1つの他のサービス・クラスを有し、前記サービス・クラスの各々が、それぞれに対して定義された性能測定を有する、前記(1)記載の方法。

(5) サーバが前記第1のサービス・クラスに追加されるべきか否かを決定するステップが、所定数のサーバを前記第1のサービス・クラスに追加することが、前記第1のサービス・クラスの前記性能測定に及ぼすプラスの効果を決定するステップと、前記所定数のサーバを前記

第1のサービス・クラスに追加することが、1つ以上の他のサービス・クラスの前記性能測定に及ぼすマイナスの効果を決定するステップと、前記第1のサービス・クラスの前記性能測定に及ぼすプラスの効果が、1つ以上の他のサービス・クラスの前記性能測定に及ぼすマイナスの効果を上回るか否かを決定するステップとを含む、前記(4)記載の方法。

(6) サーバが追加されるべき、前記クラスタ内のターゲット・システムを決定するステップが、前記クラスタ内の任意のシステムが、1つ以上の追加のサーバを追加するための十分な遊休容量を有するか否かを決定するステップと、前記クラスタ内の任意のシステムが、1つ以上の追加のサーバを追加するための十分な遊休容量を有する場合、1つのこうしたシステムを前記ターゲット・システムとして選択するステップとを含む、前記(1)記載の方法。

(7) 最大の未使用容量を有するシステムが前記ターゲット・システムとして選択される、前記(6)記載の方法。

(8) サーバが追加されるべき、前記クラスタ内のターゲット・システムを決定するステップが、前記クラスタ内のいずれのシステムも、1つ以上の追加のサーバを追加するための十分な遊休容量を有さない場合、1つ以上のサーバの追加が他の作業に最小の影響を及ぼす前記クラスタ内のシステムを決定するステップと、そのシステムを前記ターゲット・システムとして選択するステップとを含む、前記(6)記載の方法。

(9) マシンにより読出され、前記マシンにより実行されて、前記(1)記載の方法を実行する命令のプログラムを実現する、プログラム記憶装置。

(10) 情報処理システムのクラスタ内においてサーバの数を制御する装置であって、あるサービス・クラスに属する入来作業要求が、前記クラスタ全体に渡るキューに配置され、前記クラスタの前記システム上の1つ以上のサーバにより処理されるものにおいて、1つ以上のサーバが前記サービス・クラスに追加されるべきか否かを決定する手段と、1つ以上のサーバが前記サービス・クラスに追加されるべきと決定される場合、前記サーバが追加されるべき、前記クラスタ内のターゲット・システムを決定する手段と、前記サーバを前記ターゲット・システム上に追加する手段とを含む、装置。

(11) 情報処理システムのクラスタ内において、キュー内の各作業要求を処理可能なサーバの可用性を保証する方法であって、入来作業要求が前記キューに配置され、前記システム上の1つ以上のサーバにより処理されるものにおいて、前記キューに対してサーバを有さない前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在するか否かを決定するステップと、前記キューに対してサーバを有さない前記クラスタのサブセットだけに親和性を有する作業要求が、前記

キュー内に存在すると決定された場合、前記作業要求が親和性を有する前記サブセット内のシステム上で、前記キューに対してサーバを始動するステップとを含む、方法。

(12) 前記決定するステップが周期的インタバルで実行される、前記(11)記載の方法。

(13) 異なるサービス・クラスに割当てられる作業要求が、異なるキューに配置され、前記決定するステップが前記キューの各々に対して実行される、前記(11)記載の方法。

(14) 前記決定するステップが、前記クラスタ内の各システムにより実行される、前記(11)記載の方法。

(15) 前記決定するステップが、前記キューが前記システム上にサーバを有するか否かを決定するステップと、前記キューが前記システム上にサーバを有さない場合、前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在するか否かを決定するステップとを含む、前記(14)記載の方法。

(16) マシンにより読出され、前記マシンにより実行されて、前記(11)記載の方法を実行する命令のプログラムを実現する、プログラム記憶装置。

(17) 情報処理システムのクラスタ内において、キュー内の各作業要求を処理可能なサーバの可用性を保証する装置であって、入来作業要求が前記キューに配置され、前記システム上の1つ以上のサーバにより処理されるものにおいて、前記キューに対してサーバを有さない前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在するか否かを決定する手段と、前記キューに対してサーバを有さない前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在すると決定された場合、前記作業要求が親和性を有する前記サブセット内のシステム上で、前記キューに対してサーバを始動する手段とを含む、装置。

【図面の簡単な説明】

【図1】本発明のために適応化された制御オペレーティング・システム及びシステム資源マネージャ要素を有する、コンピュータ・システムを示すシステム構造図である。

【図2】ネットワークから、本発明の作業負荷マネージャにより管理されるサーバ・アドレス空間へのクライアント作業要求の流れを示す図である。

【図3】資源ボトルネックを選択するために使用される状態データを示す図である。

【図4】ボトルネック発見機能の論理フローを示すフロー

ーチャートである。

【図5】サーバの数を増加することにより、改善された性能をアクセスするステップのフローチャートである。

【図6】キュー準備完了ユーザ平均のサンプル・グラフである。

【図7】キュー遅延のサンプル・グラフである。

【図8】クラスタ内のどこかに、キュー上の各要求を実行可能な少なくとも1つのサーバが存在することを保証するプロシージャを示す図である。

【図9】サーバを始動すべきクラスタ内の最善のシステムを決定するプロシージャを示す図である。

【図10】ドナーへの影響が最も小さいシステムを見出すプロシージャを示す図である。

【符号の説明】

90 クラスタ

100 コンピュータ・システム

101 オペレーティング・システム

102 ディスパッチャ

105 作業負荷マネージャ(WLM)

106 クラス・テーブル・エントリ

107 サーバの数

110 応答時間目標

111 実行速度目標

112 システム資源マネージャ(SRM)

113 サンプル・データ

114 多重システム目標駆動型制御装置(MGDP C)

117 ボトルネック発見手段

118 固定手段

123 ドナー選択手段

124 ネット値手段

125 サンプル・データ履歴

126 応答時間履歴

141 性能目標

151 多重システム性能指標

152 ローカル性能指標

155 データ伝送機構

157 リモート応答時間履歴

158 リモート速度履歴

160 作業マネージャ

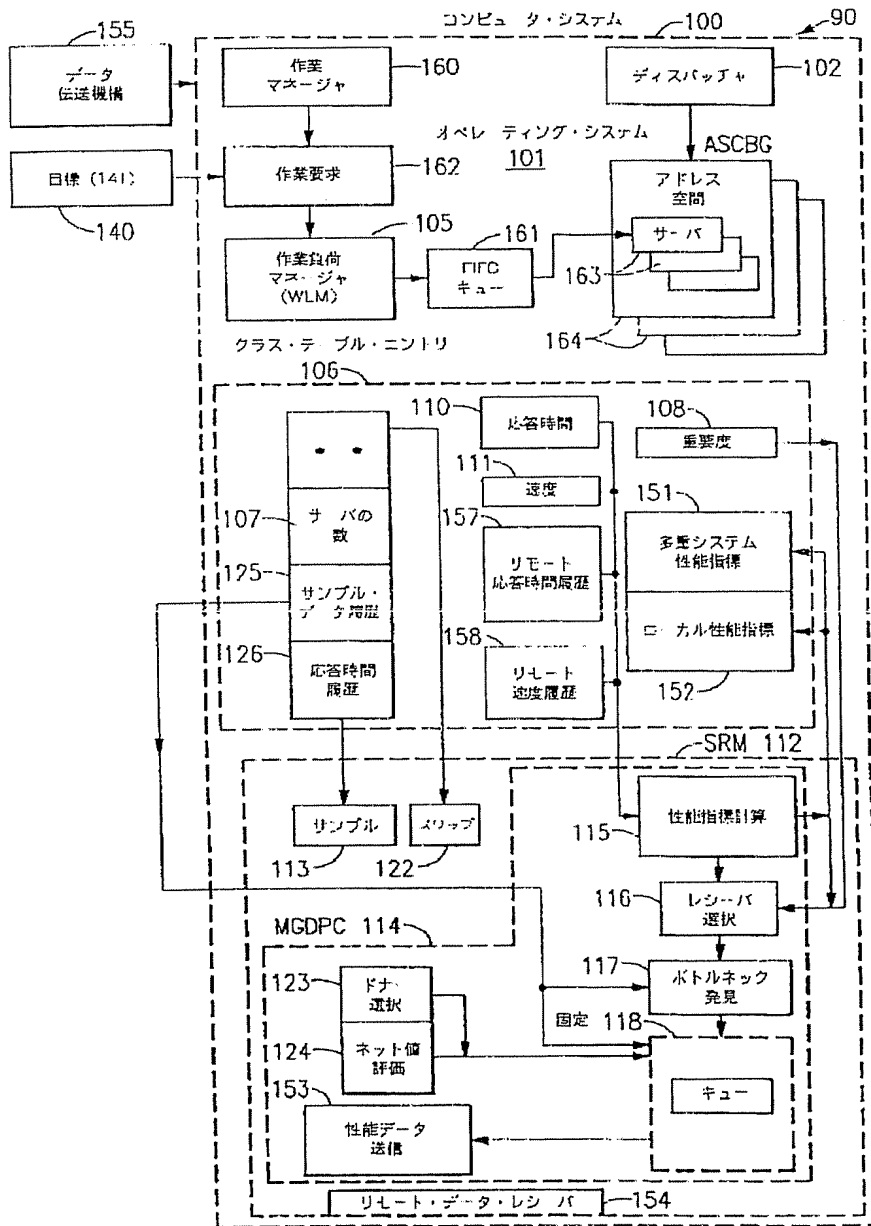
161 キュー

162 作業要求

163 サーバ

164 アドレス空間

【図 1】

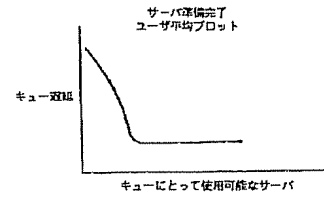


【図 3】

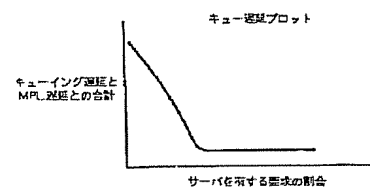
ボトルネック発見のための状態サンプル

CPU 遅延 サンプル	MPL 遅延 サンプル	スワップ 遅延 サンプル	AUX ページング 遅延 サンプル	...	キュー 遅延 サンプル
フラグ	フラグ	フラグ	フラグ	...	フラグ

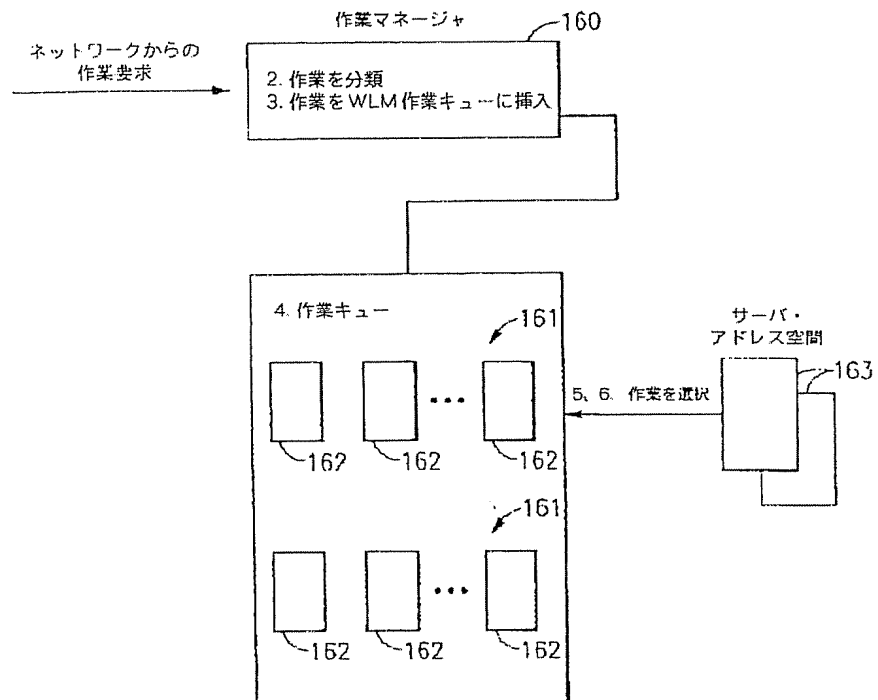
【図 6】



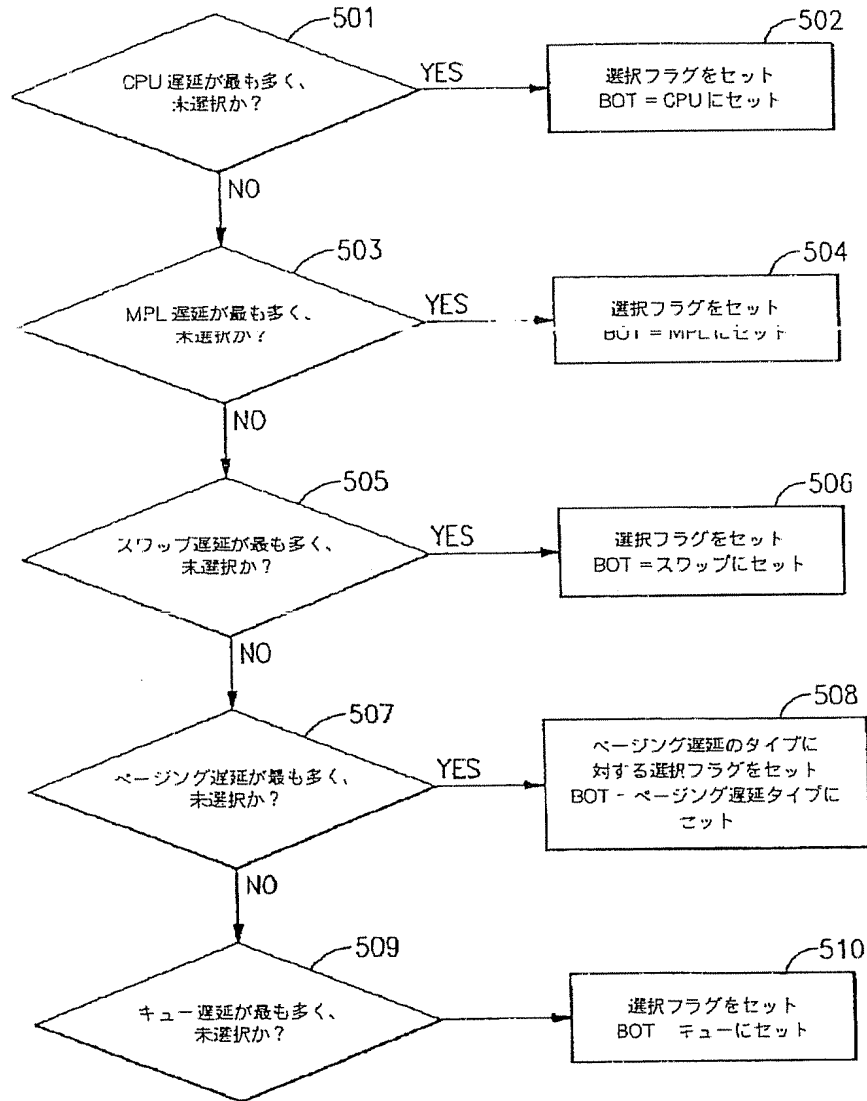
【図 7】



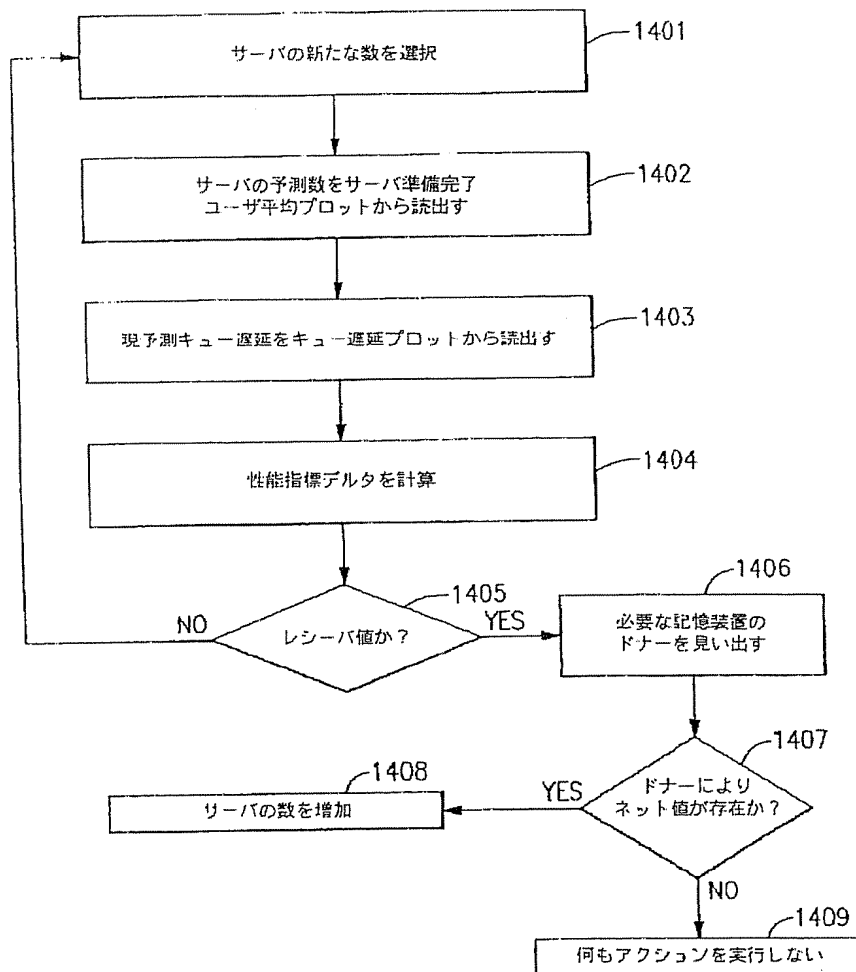
【図 2】



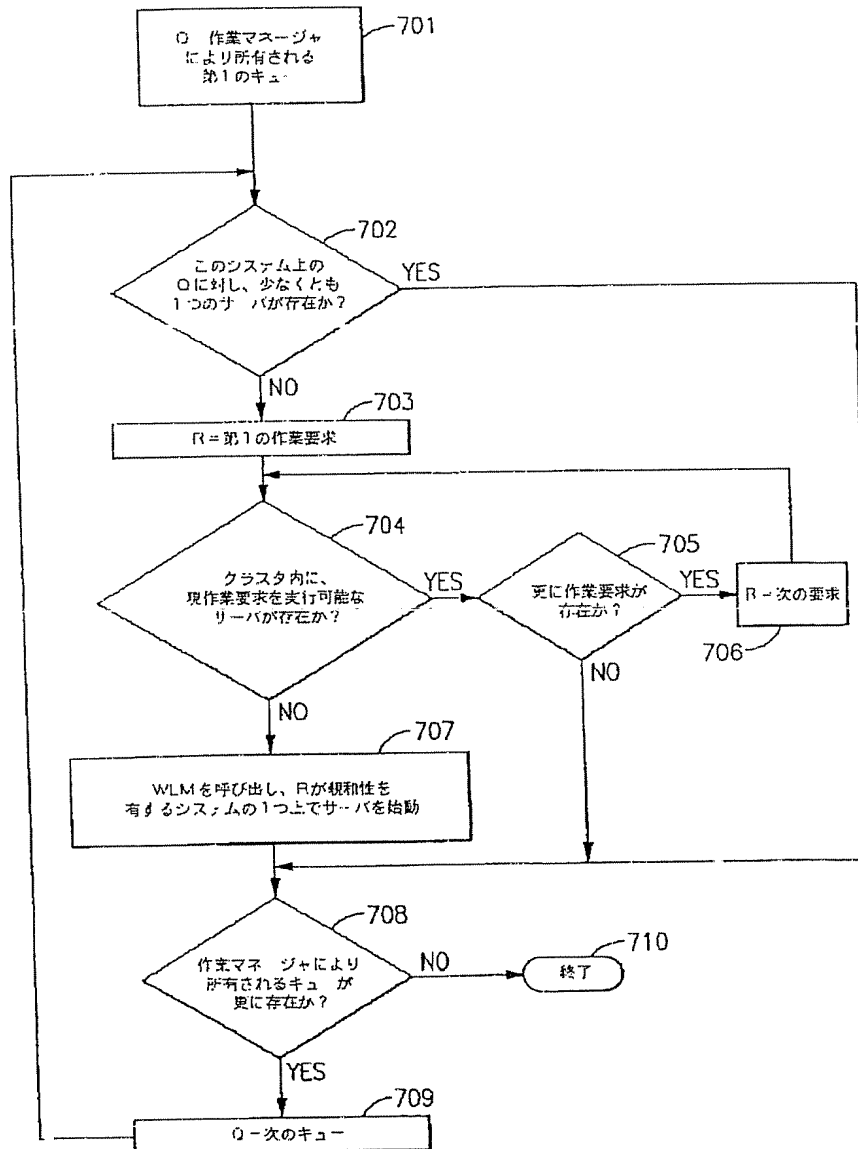
【図 4】



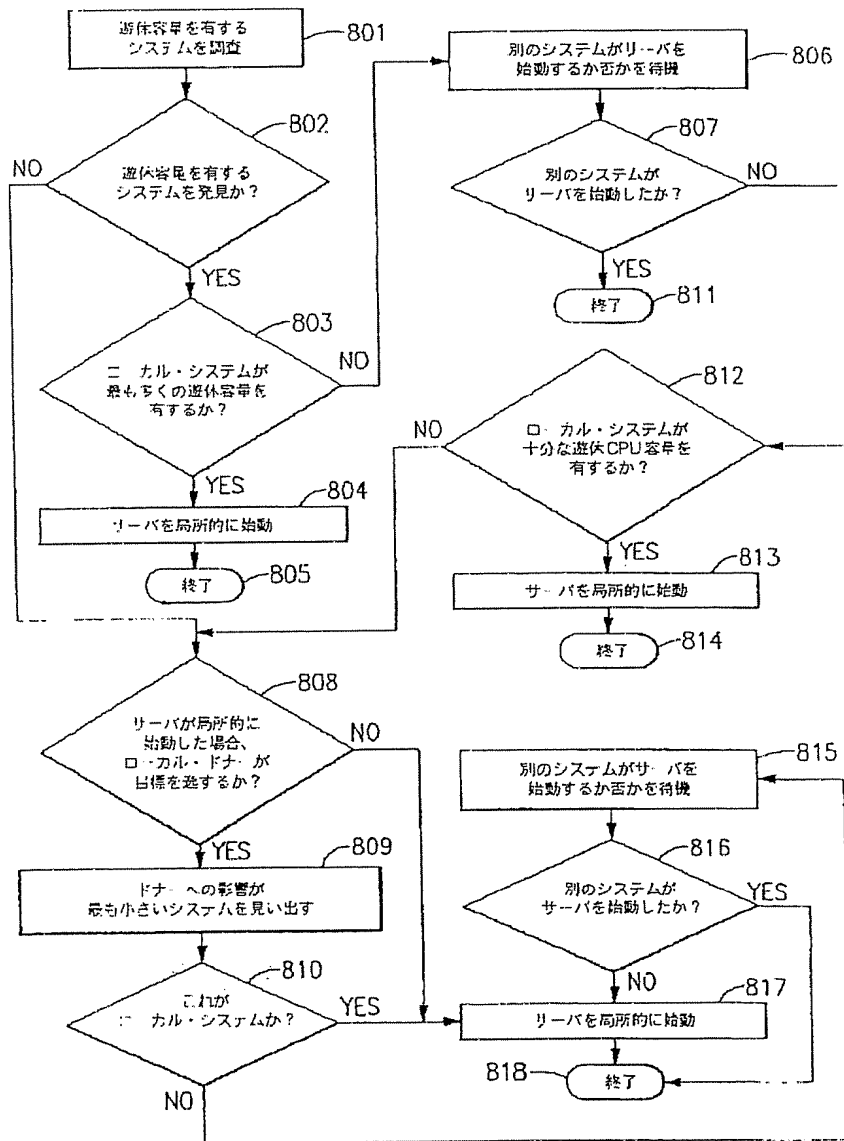
【図 5】



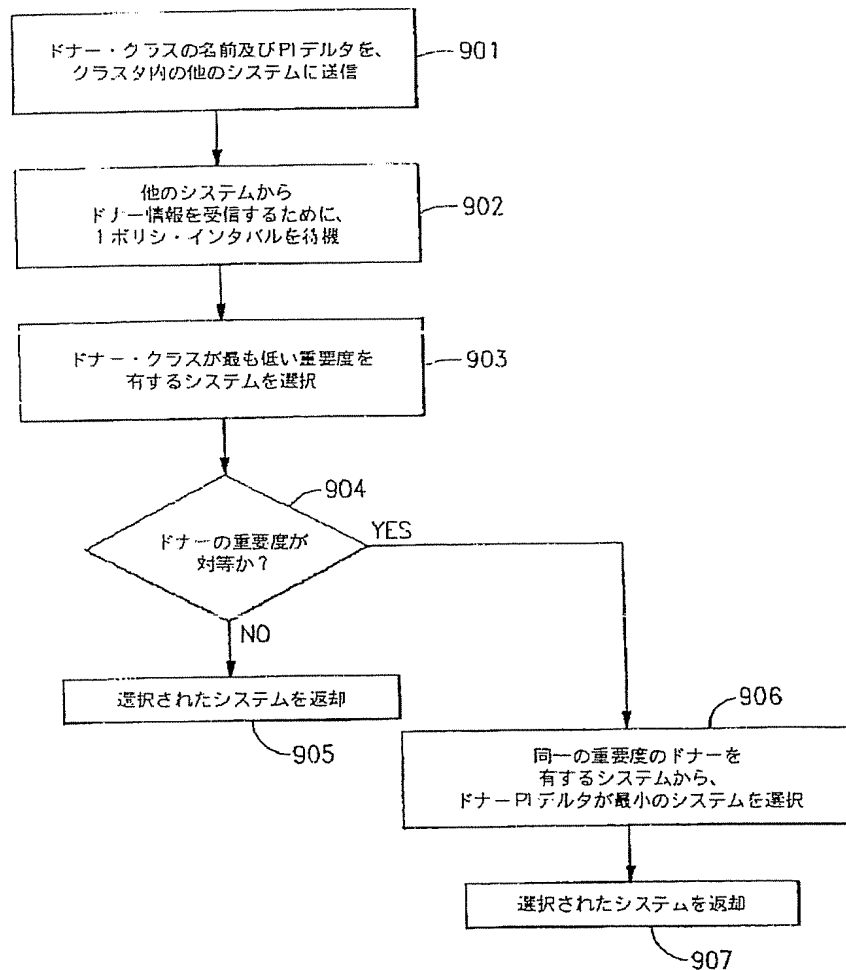
【図 8】



【図 9】



【図 10】



フロントページの続き

(72) 発明者 キャサリン・ケイ・エイラート
アメリカ合衆国12590、ニューヨーク州ワ
ッピングーズ・フォールズ、シャーウッド・
ハイツ・ドライブ 34

(72) 発明者 ジョン・イー・アーウィ
アメリカ合衆国12603、ニューヨーク州ボ
キプシ、グレンウッド・アベニュー 43